xml:tm - a radical new approach to translating XML based documents.

Andrzej Zydroń

CTO XML-INTL
PO Box 2167
Gerrards Cross
Bucks SL9 8XF
UK
azydron@xml-intl.com

Abstract

This paper describes the proposed xml:tm standard. xml:tm a revolutionary new approach to the problems of translating electronic document content. It leverages existing OASIS, W3C and LISA standards to produce a radically new view of XML documents: text memory. xml:tm has been offered to LISA OSCAR for consideration as a LISA OSCAR standard.

1. Translating XML documents

XML has become one of the defining technologies that is helping to reshape the face of both computing and publishing. It is helping to drive down costs and dramatically increase interoperability between diverse computer systems. From the localization point of view XML offers many advantages:

1. A well defined and rigorous syntax that is backed up by a rich tool set that allows documents to be validated and proven.
2. A well defined character encoding system that includes support for Unicode.
3. The separation of form and content which allows both multi target publishing (PDF, Postscript, WAP, HTML, XHTML, online help) from one source.

Companies that have adopted XML based publishing have seen significant cost savings compared with SGML or older proprietary systems. The localization industry has also enthusiastically used XML as the basis of exchange standards such as the LISA OSCAR TMX[1] (Translation Memory eXchange), TBX[2] (TermBase Exchange), SRX[3] (Segmentation Rules eXchange) standards, as well as GMX[4] (Global Information Management Metrics eXchange) set of proposed standards (Volume, Complexity and Quality). OASIS has also contributed in this field with XLIFF[5] (XML Localization Interchange File Format) and TransWS[6] (Translation Web Services). In addition the W3C ITS[7] Committee under the chair of Yves Savourel is working towards a common tag set of Elements and Attributes for Localization (Translatability of content, localization process in general etc.).

Another significant development affecting XML and localization has been the OASIS DITA (Darwin Information Technology Architecture) standard. DITA[8] provides a comprehensive architecture for the authoring, production and delivery of technical documentation. DITA was originally developed within IBM and then donated to OASIS. The essence of DITA is the concept of topic based publication construction and development that allows for the modular reuse of specific sections. Each section is authored independently and then each publication is constructed from the section modules. This means that individual sections only need to be authored and translated once, and may be reused many times over in different publications.

A core component of DITA is the concept of reuse through a well defined system for establishing a usable level of granularity within document components. DITA represents a very intelligent and well thought out approach to the process of publishing technical documentation. At the core of DITA is the concept the 'topic'. A topic is a unit of information that describes a single task, concept, or reference item. DITA uses an object orientated approach to the concept of topics encompassing the standard object oriented characteristics of polymorphism, encapsulation and message passing.

The main features of DITA are:

1. Topic centric level of granularity
2. Substantial reuse of existing assets
3. Specialization at the topic and domain level
4. Meta data property based processing
5. Leveraging existing popular element names and attributes from XHTML
6. The basic message behind DITA is reuse: 'write once, translate once, reuse many times'.

2. xml:tm

xml:tm[9] is a radical new approach to the problem of translation for XML documents. In essence it takes the DITA message of reuse and implements it at the sentence level. It does this by leveraging the power of XML to embed additional information within the XML document itself. xml:tm has additional benefits which emanate from its use. The main way it does this is through the use of the XML namespace syntax.

xml:tm was developed by XML-INTL and donated to the LISA OSCAR steering committee for consideration as a LISA OSCAR standard. In essence xml:tm is a perfect companion to DITA - the two fit together hand in glove in terms of interoperability and localization.

At the core of xml:tm is the concept of “text memory”. Text memory comprises two components:

1. Author Memory
2. Translation Memory

3. Author Memory

XML namespace is used to map a text memory view onto a document. This process is called segmentation. The text memory view works at the sentence level of granularity – the text unit. Each individual xml:tm text unit is allocated a unique identifier. This unique identifier is immutable for the life of the document. As a document goes through its life cycle the unique identifiers are maintained and new ones are allocated as required. This aspect of text memory is called author memory. It can be used to build author memory systems which can be used to simplify and improve the consistency of authoring.

The following diagram shows the how the tm namespace maps onto an existing xml document:

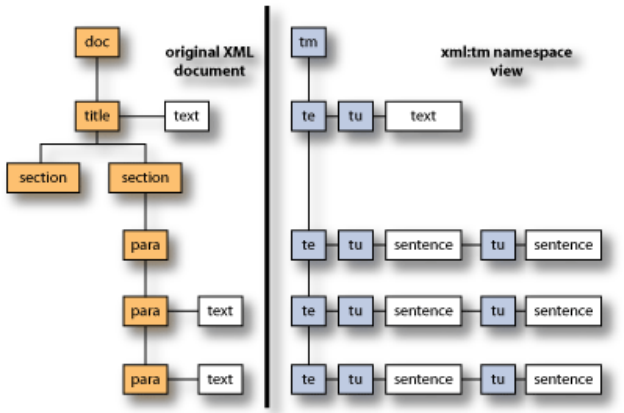


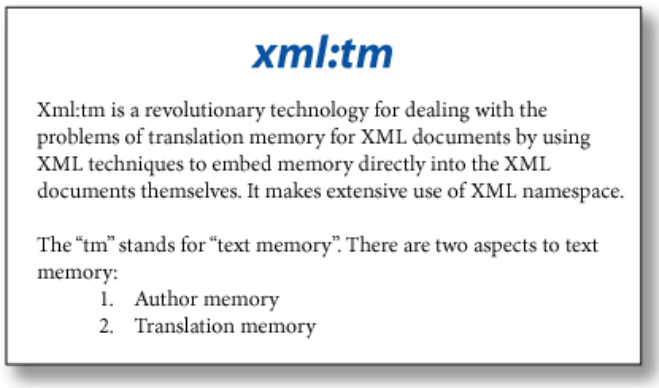
Figure 1. How xml:tm namespace maps onto an existing xml document.

In the above diagram "te" stands for "text element" (an XML element that contains text) and "tu" stands for "text unit" (a single sentence or stand alone piece of text).

The following simplified example shows how xml:tm is implemented in an XML document. The xml:tm elements are highlighted in red to show how xml:tm maps onto an existing XML document.:

```
<?xml version="1.0" encoding="UTF-8" ?>
<office:document-content
   xmlns:text="http://openoffice.org/2000/text"
   xmlns:tm="urn:xmlintl-tm-tags"
xmlns:xlink="http://www.w3.org/1999/xlink">
   <tm:tm>
     <text:p text:style-name="Text body">
       <tm:te id="e1" tuval="2">
           <tm:tu id="u1.1"> Xml:tm is a
revolutionary technology for dealing
          with the problems of translation
memory for XML documents by using
          XML techniques to embed memory
directly into the XML documents themselves.
</tm:tu>
           <tm:tu id="u1.2"> It makes extensive
use of XML namespace. </tm:tu>
       </tm:te>
     </text:p>
     <text:p text:style-name="Text body">
       <tm:te id="e2">
           <tm:tu id="u2.1"> The “tm” stands for
“text memory”. </tm:tu>
           <tm:tu id="u2.2"> There are two
aspects to text memory: </tm:tu>
       </tm:te>
     </text:p>
     <text:ordered-list text:continue-
numbering="false" text:style-name="L1">
       <text:list-item>
         <text:p text:style-name="P3">
           <tm:te id="e3">
               <tm:tu id="u3.1"> Author
memory</tm:tu>
           </tm:te>
         </text:p>
       </text:list-item>
       <text:list-item>
         <text:p text:style-name="P3">
           <tm:te id="e4">
               <tm:tu id="u4.1"> Translation
memory</tm:tu>
           </tm:te>
         </text:p>
       </text:list-item>
     </text:ordered-list>
   </tm:tm>
</office:document-content>
```

And the composed document:



xml:tm

Xml:tm is a revolutionary technology for dealing with the problems of translation memory for XML documents by using XML techniques to embed memory directly into the XML documents themselves. It makes extensive use of XML namespace.

The “tm” stands for “text memory”. There are two aspects to text memory:

1. Author memory
2. Translation memory

Figure 2. The composed document.

4. Translation Memory

When an xml:tm namespace document is ready for translation the namespace itself specifies the text that is to be translated. The tm namespace can be used to create an XLIFF document for translation.

4.1. XLIFF

XLIFF[5] is another XML format that is optimized for translation. Using XLIFF you can protect the original document syntax from accidental corruption during the translation process. In addition you can supply other relevant information to the translator such as translation memory and preferred terminology.

The following is an example of an XLIFF document based on the previous example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliff PUBLIC "-//XML-INTL XLIFF-XML
1.0//EN" "file:xliff.dtd">
<xliff version="1.0">
  <file datatype="xml" source-language="en-USA"
target-language="es-ESP">
   <header>
    <count-group name="Totals">
     <count count-type="TextUnits"
unit="transUnits">40</count>
     <count count-type="TotalWordCount"
unit="words">416</count>
    </count-group>
   </header>
   <body>
     <trans-unit id="t1">
        <source> xml:tm</source>
        <target> xml:tm </target>
     </trans-unit>
     <trans-unit id="t2">
        <source> Xml:tm is a revolutionary
technique for dealing with the problems of
translation memory for XML documents by using
XML techniques and embedding memory directly
into the XML documents themselves.
        </source>
        <target> Xml:tm is a revolutionary
technique for dealing with the problems of
translation memory for XML documents by using
XML techniques and embedding memory directly
into the XML documents themselves.
        </target>
     </trans-unit>
     <trans-unit id="t3">
        <source> It makes extensive use of XML
namespace.
        </source>
        <target> It makes extensive use of XML
namespace.
        </target>
     </trans-unit>
     <trans-unit id="t4">
        <source> The “tm” stands for “text
memory”. </source>
        <target> The “tm” stands for “text
memory”. </target>
     </trans-unit>
     <trans-unit id="t5">
        <source> There are two aspects to text
memory: </source>
        <target> There are two aspects to text
memory: </target>
     </trans-unit>
     <trans-unit id="t6">
        <source> Author memory </source>
        <target> Author memory </target>
     </trans-unit>
     <trans-unit id="t7">
        <source> Translation memory </source>
        <target> Translation memory </target>
     </trans-unit>
    </body>
  </file>
</xliff>
```

The magenta colored text signifies where the translated text will replace the source language text as shown below:

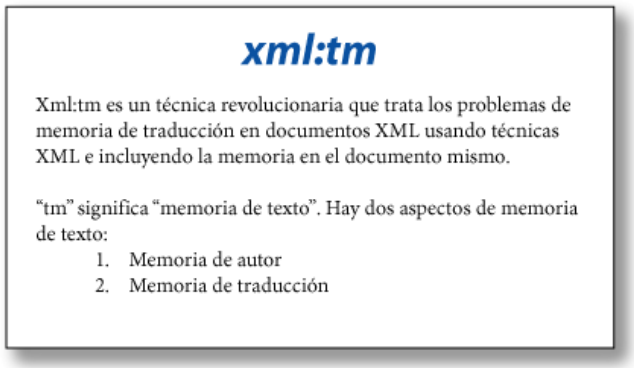
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliff PUBLIC "-//XML-INTL XLIFF-XML
1.0//EN" "file:xliff.dtd">
<xliff version="1.0">
  <file datatype="xml" source-language="en-USA"
target-language="es-ESP">
   <header>
    <count-group name="Totals">
     <count count-type="TextUnits"
unit="transUnits">40</count>
     <count count-type="TotalWordCount"
unit="words">416</count>
    </count-group>
   </header>
   <body>
     <trans-unit id="t1">
        <source> xml:tm</source>
        <target> xml:tm </target>
     </trans-unit>
     <trans-unit id="t2">
        <source> Xml:tm is a revolutionary
technique for dealing with the problems of
translation memory for XML documents by using
XML techniques and embedding memory directly
into the XML documents themselves.
        </source>
        <target> Xml:tm es un técnica
revolucionaria que trata los problemas de
memoria de traducción en documentos XML usando
técnicas XML e incluyendo la memoria en el
documento mismo.
        </target>
     </trans-unit>
     <trans-unit id="t3">
        <source> It makes extensive use of XML
namespace.
        </source>
        <target> E sta técnica hace extensor uso
de XML namespace.
        </target>
     </trans-unit>
     <trans-unit id="t4">
        <source> The “tm” stands for “text
memory”. </source>
        <target> “tm” significa “memoria de
texto”. </target>
     </trans-unit>
     <trans-unit id="t5">
        <source> There are two aspects to text
memory: </source>
        <target> Hay dos aspectos de memoria de
texto: </target>
     </trans-unit>
     <trans-unit id="t6">
        <source> Author memory </source>
        <target> Memoria de autor </target>
     </trans-unit>
     <trans-unit id="t7">
        <source> Translation memory </source>
        <target> Memoria de traducción </target>
     </trans-unit>
    </body>
  </file>
</xliff>
```

When the translation has been completed the target language text can be merged with the original document to create a new target language version of that document. The net result is a perfectly aligned source and target language document.

The following is the translated xml:tm document in Spanish:

```
<?xml version="1.0" encoding="UTF-8" ?>
<office:document-content
   xmlns:text="http://openoffice.org/2000/text"
   xmlns:tm="urn:xmlintl-tm-tags"
xmlns:xlink="http://www.w3.org/1999/xlink">
   <tm:tm>
     <text:p text:style-name="Text body">
       <tm:te id="e1" tuval="2">
           <tm:tu id="u1.1"> Xml:tm es un
          técnica revolucionaria que trata los
problemas de memoria de
          traducción en documentos XML usando
técnicas XML e
          incluyendo la memoria en el documento
mismo. </tm:tu>
           <tm:tu id="u1.2"> E sta técnica hace
extensor uso de XML namespace. </tm:tu>
       </tm:te>
     </text:p>
     <text:p text:style-name="Text body">
       <tm:te id="e2">
           <tm:tu id="u2.1"> “tm” significa
“memoria de texto”. </tm:tu>
           <tm:tu id="u2.2"> Hay dos aspectos de
memoria de texto: </tm:tu>
       </tm:te>
     </text:p>
     <text:ordered-list text:continue-
numbering="false" text:style-name="L1">
       <text:list-item>
         <text:p text:style-name="P3">
           <tm:te id="e3">
                <tm:tu id="u3.1"> Memoria de
autor</tm:tu>
           </tm:te>
         </text:p>
       </text:list-item>
       <text:list-item>
         <text:p text:style-name="P3">
           <tm:te id="e4">
                <tm:tu id="u4.1"> Memoria de
traducción</tm:tu>
           </tm:te>
         </text:p>
       </text:list-item>
     </text:ordered-list>
   </tm:tm>
</office:document-content>
```

This is an example of the composed translated text:



xml:tm

Xml:tm es un técnica revolucionaria que trata los problemas de memoria de traducción en documentos XML usando técnicas XML e incluyendo la memoria en el documento mismo.

“tm” significa “memoria de texto”. Hay dos aspectos de memoria de texto:

1. Memoria de autor
2. Memoria de traducción

Figure 3. The composed translated document.

The source and target text is linked at the sentence level by the unique xml:tm identifiers. When the document is revised new identifiers are allocated to modified or new text units. When extracting text for translation of the updated source document the text units that have not changed can be automatically replaced with the target language text. The resultant XLIFF file will look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliff PUBLIC "-//XML-INTL XLIFF-XML
1.0//EN" "file:xliff.dtd">
<xliff version="1.0">
  <file datatype="xml" source-language="en-USA"
target-language="es-ESP">
   <header>
    <count-group name="Totals">
     <count count-type="TextUnits"
unit="transUnits">40</count>
     <count count-type="TotalWordCount"
unit="words">416</count>
    </count-group>
   </header>
   <body>
     <trans-unit translate="no" id="t1">
        <source> xml:tm</source>
        <target state-qualifier="exact-matched">
xml:tm </target>
     </trans-unit>
     <trans-unit translate="no" id="t2">
        <source> Xml:tm is a revolutionary
technique for dealing with the problems of
translation memory for XML documents by using
XML techniques and embedding memory directly
into the XML documents themselves.
        </source>
        <target state-qualifier="exact-matched">
Xml:tm es un técnica revolucionaria que trata
los problemas de memoria de traducción en
documentos XML usando técnicas XML e incluyendo
la memoria en el documento mismo.
        </target>
     </trans-unit>
     <trans-unit translate="no" id="t3">
        <source> It makes extensive use of XML
namespace.
        </source>
        <target state-qualifier="exact-matched">
E sta técnica hace extensor uso de XML
namespace.
        </target>
     </trans-unit>
```

```
     <trans-unit translate="no" id="t4">
        <source> The “tm” stands for “text
memory”. </source>
        <target state-qualifier="exact-matched">
“tm” significa “memoria de texto”. </target>
     </trans-unit>
     <trans-unit translate="no" id="t5">
        <source> There are two aspects to text
memory: </source>
        <target state-qualifier="exact-matched">
Hay dos aspectos de memoria de texto: </target>
     </trans-unit>
     <trans-unit translate="no" id="t6">
        <source> Author memory </source>
        <target state-qualifier="exact-matched">
Memoria de autor </target>
     </trans-unit>
     <trans-unit translate="no" id="t7">
        <source> Translation memory </source>
        <target state-qualifier="exact-matched">
Memoria de traducción </target>
     </trans-unit>
    </body>
  </file>
</xliff>
```

4.2. Exact Matching

The matching described in the previous section is called “exact” matching. Because xml:tm memories are embedded within an XML document they have all the contextual information that is required to precisely identify text units that have not changed from the previous revision of the document. Unlike leveraged matches, perfect matches do not require translator intervention, thus reducing translation costs.

The following diagram shows how Exact Matching is achieved:

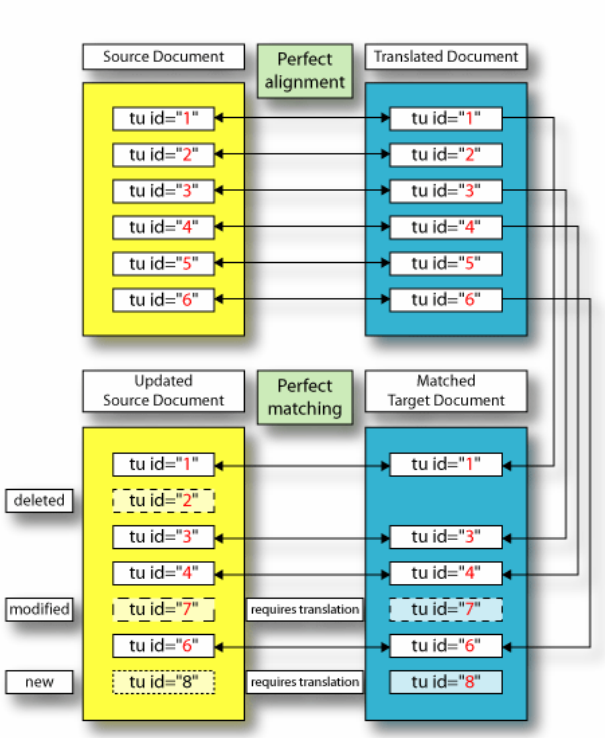


Figure 4. Exact Matching.

4.3. Matching with xml:tm

xml:tm provides much more focused types of matching than traditional translation memory systems. The following types of matching are available:

1. Exact matching

   Author memory provides exact details of any changes to a document. Where text units have not been changed for a previously translated document we can say that we have a “Exact match”. The concept of Exact Matching is an important one. With traditional translation memory systems a translator still has to proof each match, as there is no way to ascertain the appropriateness of the match. Proofing has to be paid for – typically at 60% of the standard translation cost. With Exact Matching there is no need to proof read, thereby saving on the cost of translation.

2. In document leveraged matching

   xml:tm can also be used to find in-document leveraged matches which will be more appropriate to a given document than normal translation memory leveraged matches.

3. Leveraged matching

   When an xml:tm document is translated the translation process provides perfectly aligned source and target language text units. These can be used to create traditional translation memories, but in a consistent and automatic fashion.

4. In document fuzzy matching

   During the maintenance of author memory a note can be made of text units that have only changed slightly. If a corresponding translation exists for the previous version of the source text unit, then the previous source and target versions can be offered to the translator as a type of close fuzzy match.

5. Fuzzy matching

   The text units contained in the leveraged memory database can also be used to provide fuzzy matches of similar previously translated text. In practice fuzzy matching is of little use to translators except for instances where the text units are fairly long and the differences between the original and current sentence are very small.

6. Non translatable text

In technical documents you can often find a large number of text units that are made up solely of numeric, alphanumeric, punctuation or measurement items. With xml:tm these can be identified during authoring and flagged as non translatable, thus reducing the word counts. For numeric and measurement only text units it is also possible to automatically convert the decimal and thousands designators as required by the target language.

5. xml:tm and other Localization Industry Standards

xml:tm was designed from the outset to integrate closely with and leverage the potential of other relevant XML based Localization Industry Standards.

In particular:

1. SRX[3] (Segmentation Rules eXchange)

   xml:tm mandates the use of SRX for text segmentation of paragraphs into text units.

2. Unicode Standard Annex #29[11] Text Boundaries

   xml:tm mandates the use of Unicode Standard Annex #29 for tokenization of text into words.

3. XLIFF[5] (XML Localization Interchange File Format)

   xml:tm mandates the use of XLIFF for the actual translation process. xml:tm is designed to facilitate the automated creation of XLIFF files from xml:tm enabled documents, and after translation to easily create the target versions of the documents.

4. GMX-V[4] (Global Information Management Metrics eXchange - Volume)

   xml:tm mandates the use of GMX-V for all metrics concerning authoring and translation.

5. DITA[8] (Darwin Information Technology Architecture)

   xml:tm is a perfect match for DITA, taking the DITA reuse principle down to sentence level.

6. TMX[1] (Translation Memory eXchange)

   xml:tm facilitates the easy creation of TMX documents, aligned at the sentence level.

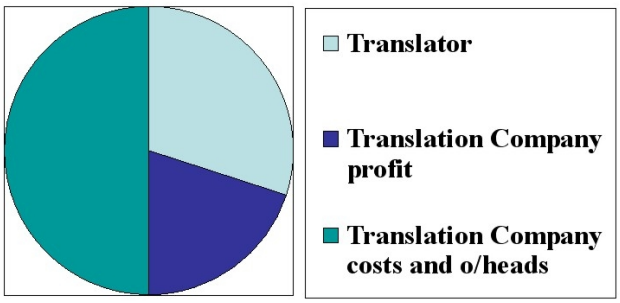
6. Controlling Matching and Word counts

You can use xml:tm to create an integrated and totally automated translation environment. The presence of xml:tm allows for the automation of what would otherwise be labour intensive processes. The previously translated target version of the document serves as the basis for the exact matching of unchanged text. In addition xml:tm allows for the identification of text that does not require translation (text units comprising solely punctuation or numeric or alphanumeric only text) as well as providing for in-document leveraged and fuzzy matching.

In essence xml:tm has already pre-prepared a document for translation and provided all of the facilities to produce much more focused matching. After exhausting all of the in-document matching possibilities any unmatched xml:tm text units can be searched for in the traditional leveraged and fuzzy search manner.

The presence of xml:tm can be used to totally automate the extraction and matching process. This means that the customer is in control of all of the translation memory matching and word count processes, all based on open standards. This not only substantially reduces the cost of preparing the document for translation, which is usually charged for by localization service providers, but is also much more efficient and cost effective as it is totally automated. The customer now controls the translation memory matching process and the word counts.

In a study conducted in 2002 by the Localization Research Centre the typical cost of the actual translation accounted for only 33% of the cost of localization for a typical project. Over 50% of the cost was consumed by administrative and project management charges. With xml:tm in an automated translation environment you can substantially reduce the costs of translation.



Source Professor Reinhard Schäler LRC - ASLIB 2002

Figure 5. The true costs of a traditional translation process.

The output from the text extraction process can be used to generate automatic word and match counts by the customer. This puts the customer in control of the word counts, rather than the supplier. This is an important distinction and allows for a tighter control of costs.

Traditional translation scenario:

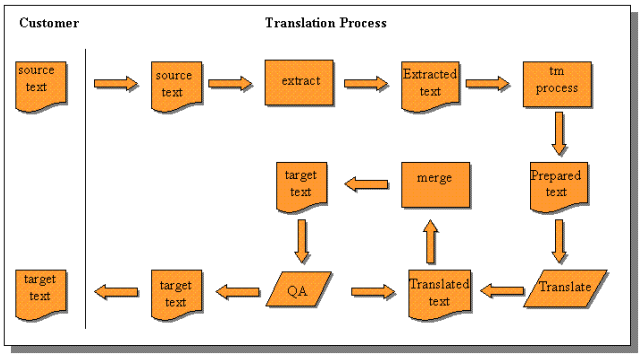


Figure 6. Traditional translation scenario.

In the xml:tm translation scenario all processing takes place within the customer's environment:

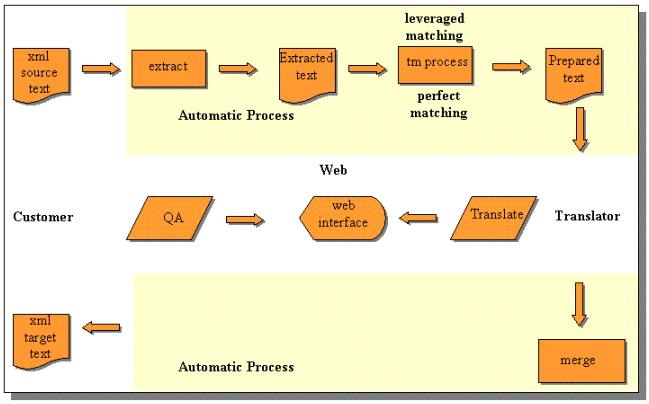


Figure 7. xml:tm translation scenario.

7. On line translation.

xml:tm mandates the use of XLIFF as the exchange format for translation. XLIFF format can be used to create dynamic web pages for translation. A translator can access these pages via a browser and undertake the whole of the translation process over the Internet. This has many potential benefits. The problems of running filters and the delays inherent in sending data out for translation such as inadvertent corruption of character encoding or document syntax, or simple human work flow problems can be totally avoided. Using XML technology it is now possible to both reduce and control the cost of translation as well as reduce the time it takes for translation and improve the reliability.

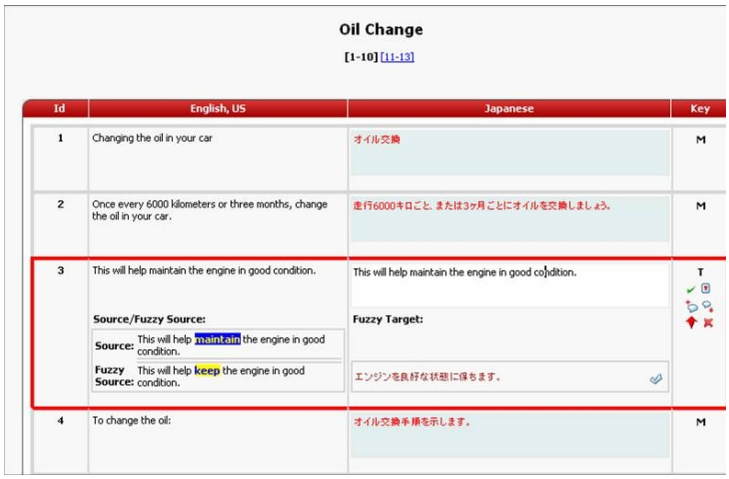


Figure 8. An example of a web based translator environment:

8. Benefits of using xml:tm

The following is a list of the main benefits of using the xml:tm approach to authoring and translation:

1. The ability to build consistent authoring systems.
2. Automatic production of authoring statistics.
3. Automatic alignment of source and target text.
4. Aligned texts can be used to populate leveraged matching tm database tables.
5. Exact translation matching for unchanged text units.
6. In-document leveraged and modified text unit matching.
7. Automatic production of word count statistics.
8. Automatic generation of exact, leveraged, previous modified or fuzzy matching.
9. Automatic generation of XLIFF files.
10. Protection of the original document structure.
11. The ability to provide on line access for translators.
12. Can be used transparently for relay translation.
13. An open standard that is based and interoperates with other relevant open standards (SRX[3], Unicode TR29[11], XLIFF[5], TMX[1], GMX-V[4]).

9. Summary

xml:tm is a namespace based technology created and maintained by XML-INTL based on XML and Localization Industry Standards for the benefit of the translation and authoring communities. Full details of the xml:tm definitions (XML Data Type Definition and XML Schema) are available from the XML-INTL web site (http://www.xml-intl.com).

The xml:tm approach reduces translation costs in the following ways:

1. Translation memory is held by the customer within the documents.

2. Exact Matching reduces translation costs by eliminating the need for translators to proof these matches.
3. Translation memory matching is much more focused than is the case with traditional translation memory systems providing better results.
4. It allows for relay translation memory processing via an intermediate language.
5. All translation memory, extraction and merge processing is automatic, there is no need for manual intervention.
6. Translation can take place directly via the customer's web site.
7. All word counts are controlled by the customer.
8. The original XML documents are protected from accidental damage.
9. The system is totally integrated into the XML framework, making maximum use of the capabilities of XML to address authoring and translation.

10. References

[1] TMX - Translation Memory eXchange format : *http://www.lisa.org/tmx/*

[2] TBX - TermBase eXchange format : *http://www.lisa.org/tbx/*

[3] SRX - Segmentation Rules eXchange format : *http://www.lisa.org/oscar/seg/*

[4] GMX - Global Information management Metrics *: http://www.lisa.org/standards/gmx/*

[5] XLIFF - XML Localisation Interchange File Format *: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff*

[6] Translation Web Services *: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=trans-ws*

[7] W3C ITS *: html/www.w3.org/International/its*

[8] DITA - Darwin Information Technology Architecture *: html/www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita*

[9] xml:tm - detailed specification *: http://www.xml-intl.com/docs/specification/xml-tm.html*

[10] The Localisation Research Centre (LRC) *: http://www.localisation.ie/*

[11] Unicode Standard Annex #29 *: http://www.unicode.org/reports/tr29/*